**LISTING OF CLAIMS**

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Previously Presented) In a multithreaded computing environment, a method of processing computing tasks, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

associating each task queue with a single respective worker thread;

assigning a task to a task queue in an essentially random fashion; and

from a worker thread, processing a task from a task queue not associated with the thread.

2. (Original) The method of claim 1 wherein assigning a task comprises selecting an empty task queue.

3. (Previously Presented) The method of claim 2 wherein selecting comprises determining whether a selected task queue is in a busy state.

4. (Previously Presented) The method of claim 1 further comprising, from a worker thread, processing a task from an associated task queue.

5. (Canceled)

6. (Previously Presented) In a multithreaded computing environment, a method of processing computing threads, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

associating each task queue with a single respective worker thread;

assigning a task to an assigned task queue; and

in a worker thread not associated with the assigned task queue, processing the task.

7. (Original) The method of claim 6 where assigning comprises selecting the assigned task queue based on an essentially random number.

8. (Original) The method of claim 6 wherein assigning comprises selecting an empty task queue.

9. (Original) The method of claim 8 wherein selecting comprises determining whether the task queue is in a busy state.

10. (Previously Presented) In a multithreaded computing environment, a system for processing tasks, comprising:

a plurality of worker threads, each thread capable of processing a task;

a plurality of task queues, each task queue capable of queuing a plurality of tasks and each task queue associated with a single respective worker thread;

a task scheduler for assigning a task to a task queue in an essentially random fashion; and

a worker thread processing a task from a task queue not associated with the thread.

11. (Original) The system of claim 10 wherein the task scheduler selects an empty task queue for assigning the task.

12. (Previously Presented) The system of claim 11 wherein the task scheduler further determines whether a selected task queue is in a busy state.

13. (Previously Presented) The system of claim 10 further comprising a worker thread processing a task from an associated task queue.

14. (Canceled)

15. (Original) In a multithreaded computing environment, a system for processing computing threads, comprising:

a plurality of worker threads, each thread capable of processing a task;

a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads and each task queue associated with a respective worker thread;

a task scheduler for assigning a task to an assigned task queue; and

wherein the assigned task is processed by a thread not associated with the assigned task queue.

16. (Previously Presented) The system of claim 15 wherein the task scheduler selects the assigned task queue based on an essentially random number.

17. (Original) The system of claim 15 wherein the task scheduler selects an empty task queue for assigning the task.

18. (Original) The system of claim 17 wherein the task scheduler farther determines whether the task queue is in a busy state.

19. (Previously Presented) An article of manufacturing, comprising:

a computer-readable medium;

a computer implemented program for processing computing tasks in a multithreaded computing environment embodied in the medium, the comprising instructions for:

defining a plurality of worker threads, each thread capable of processing a

task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

associating each task queue with a single respective worker thread;

assigning a task to a task queue in an essentially random fashion; and

processing, in a worker thread, a task from a task queue not associated with the thread.

20. (Original) The article of claim 19 wherein the instructions for assigning a task comprise selecting an empty task queue.

21. (Previously Presented) The article of claim 20 wherein the instructions for selecting comprise determining whether a selected task queue is in a busy state.

22. (Previously Presented) The article of claim 19 further comprising instructions for processing, in a worker thread, a task from an associated task queue.

23. (Canceled)

24. (Previously Presented) An article of manufacture, comprising:

a computer-readable medium;

a computer-implemented program for processing computing threads, in a multithreaded computing environment embodied in the medium, the program comprising instructions for:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

associating each task queue with a single respective worker thread;

assigning a task to an assigned task queue; and

in a worker thread not associated with the assigned task queue, processing the task.

25. (Original) The article of claim 24 where the instructions for assigning comprise selecting the assigned task queue based on an essentially random number.

26. (Previously Presented) The article of claim 24 wherein the instructions for assigning comprise selecting an empty task queue.

27. (Previously Presented) The article of claim 26 wherein the instructions for selecting comprise determining whether the task queue is in a busy state.

28. (Previously Presented) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a respective worker thread;

means for assigning a task to a task queue in an essentially random fashion; and

from a worker thread, means for processing a task from a task queue not associated with the thread.

29. (Previously Presented) The system of claim 28 wherein the means for assigning a task comprises means for selecting an empty task queue.

30. (Previously Presented) The system of claim 29 wherein the means for selecting comprises determining whether a selected task queue is in a busy state.

31. (Previously Presented) The system of claim 28 further comprising, from a worker thread, means for processing a task from an associated task queue.

32. (Canceled)

33. (Previously Presented) In a multithreaded computing environment, a method of processing computing tasks, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

associating each task queue with a single respective worker thread;

assigning a task to an empty task queue in an essentially random fashion; and

from a worker thread, processing a task from a task queue not associated with the thread.

34. (Previously Presented) The method of claim 33 wherein assigning a task comprises selecting an empty task queue.

35. (Previously Presented) The method of claim 34 wherein selecting comprises determining whether a selected task queue is in a busy state.

36. (Previously Presented) The method of claim 33 further comprising, from a worker thread, processing a task from an associated task queue.

37. (Previously Presented) In a multithreaded computing environment, a method of processing computing tasks, comprising:

defining a plurality of worker threads, each thread capable of processing a task;

defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

associating each task queue with a single respective worker thread, an

associated task queue capable of storing tasks assigned to an associated worker thread;

assigning a task to a task queue in an essentially random fashion, comprising:

using a random number generator to identify an initial task queue;

upon determining that the initial task queue is not empty, searching the other task queues for an empty queue; and

upon finding an empty task queue, storing the task in the empty task queue; and

from a worker thread, processing a task from the associated task queue.

38. (Previously Presented) In a computer, a system for processing computing threads, comprising:

a plurality of worker thread, each thread capable of processing a task;

a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads and each task queue associated with a single respective worker thread, the associated task queue capable of storing tasks assigned to the associated worker thread;

a task scheduler for assigning a task to an assigned task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queues, searching the other task queues for an empty queue, upon determining that the initial task queue is not empty, and upon finding an empty task queue, stores the task in the empty task queue, the worker thread processing a task from the associated queue.

39. (Previously Presented) In a multithreaded computing environment, a system for processing computing threads, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks accessible by the worker threads;

means for associating each task queue with a single respective worker thread;

means for assigning a task to an assigned task queue; and

in a worker thread not associated with the assigned task queue, means for processing the task.

40. (Previously Presented) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a single respective worker thread;

means for assigning a task to an empty task queue in an essentially random fashion; and

from a worker thread, means for processing a task from a task queue not associated with the thread.

41. (Previously Presented) In a multithreaded computing environment, a system for processing computing tasks, comprising:

means for defining a plurality of worker threads, each thread capable of processing a task;

means for defining a plurality of task queues, each task queue capable of queuing a plurality of tasks;

means for associating each task queue with a single respective worker thread, an associated task queue capable of storing tasks assigned to an associated worker thread;

means for assigning a task to a task queue in an essentially random fashion, comprising:

means for using a random number generator to identify an initial task queue;

upon determining that the initial task queue is not empty, means for searching the other task queues for an empty queue; and

upon finding an empty task queue, means for storing the task in the empty task queue; and

from a worker thread, means for processing a task from the associated task queue.

42. (Previously Presented) The method of claim 37 wherein processing comprises:

from a worker thread, processing a task from a task queue not associated with the thread.

43. (Previously Presented) The system of claim 38 wherein a task scheduler comprises:

a task scheduler for assigning a task to an assigned task queue in an essentially random fashion, the task scheduler using a random number generator to identify an initial task queues, searching the other task queues for an empty queue, upon determining that the initial task queue is not empty, and upon finding an empty task queue, stores the task in the empty task queue, the worker thread processing a task not associated with the assigned task queue.

44. (Previously Presented) The system of claim 41 wherein means for processing comprises:

from a worker thread, means for processing a task from a task queue not associated with the thread.